Group: Happy happy happy

DEMO link: https://www.youtube.com/watch?v=awB8Dq4nfYI

Implementation

Locate touch spot

- Split RGB channels.
- Perform thresholding to each channel and get the final result using bitwise operation.
- Find contours by cv2.findContours(), and pick the largest contour.
- Find the centroid of contour, i.e. the touch spot, by cv2.moments()

Track touch spot

- Create a deque with max size to keep the coordinates of the set of latest touch spots.
- Filter out contour with an enclosing circle of radius less than 10 to eliminate noise.
- If the touchpad is idle for 1 second, create a new white image file and use cv2.line() to connect all the adjacent spots in order to draw the digit on the new image.
- For the bonus part, we implement real-time recognition by generating image with existing coordinates (and make prediction) at every frame

Recognize handwritten digit

- Use sklearn package to train a svc classifier with 1000 handwritten digit samples from mnist dataset
- Preprocess the image to satisfy the requirement of the training model: image size normalized to fit in a 20x20 pixel box and centered in a 28x28 image using the center of mass.
- Feed the image to the classifier to predict the digit.

What we have learnt

- Mechanism of FTIR
- Image processing with openCV
- Machine learning with sklearn

Improvement of device

- We use black tape to wrap the red LED to avoid the camera affected by the direct exposure to the LED light (which is noise)

References:

- Handwritten digit recognition on MNIST dataset using python
- Ball Tracking with OpenCV
- Recognizing HandWritten Digits in Scikit Learn
- Tensorflow, MNIST and your own handwritten digits